13

Using Fonts With X

Most X clients let you specify the font used to display text in the window, in menus and labels, or in other text fields. For example, you can choose the font used for the text in *fvwm* menus or in *xterm* windows. This chapter gives you the information you need to be able to do that. It describes what you need to know to select display fonts for use with X client applications. Some of the topics the chapter covers are:

- The basic characteristics of a font.
- The font-naming conventions and the logical font description.
- The use of wildcards and aliases for simplifying font specification
- The font search path.
- The use of a font server for accessing fonts resident on other systems on the network.
- The utilities available for managing fonts.
- The use of international fonts and character sets.
- The use of TrueType fonts.

Font technology suffers from a tension between what printers want and what display monitors want. For example, printers have enough intelligence built in these days to know how best to handle outline fonts. Sending a printer a bitmap font bypasses this intelligence and generally produces a lower quality printed page. With a monitor, on the other hand, anything more than bitmapped information is wasted. Also, printers produce more attractive print with variable-width fonts. While this is true for monitors as well, the utility of monospaced fonts usually overrides the aesthetic of variable width for most contexts in which we're looking at text on a monitor. This chapter is primarily concerned with the use of fonts on the screen.

X has a fairly complex font-naming system (which, like most things about X, is designed for maximum flexibility rather than for simplicity or ease of use). Desktop environments such as GNOME and KDE provide their own tools for managing fonts, but in a standard window-manager environment, you are generally limited to selecting fonts via command-line options or resource specifications. This wouldn't be so bad if a typical font name weren't mind-bending at first glance. Imagine typing a command line like the following every time you want to start an *xterm* window with a 20-point constant-width font:

```
xterm -fn -misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1
```

Fortunately, you can use asterisks as wildcards to simplify this name to something more reasonable:

```
xterm -fn '*fixed-medium-r-*-200-*'
```

You can also define aliases to simplify the name further, so that you might end up with a command line similar to this:

```
xterm -fn 20p
```

The fonts are stored in files in subdirectories of the */usr/X11R6/lib/X11/fonts* directory. The directory for a standard installation of X might look like this:

```
ls –p /usr/X11R6/lib/X11/fonts/
100dpi/     PEX/        Type1/         misc/
75dpi/      Speedo/     cyrillic/
```

The *misc*, *75dpi*, and *100dpi* directories contain bitmap fonts, while the *Speedo* and *Type1* directories contain scalable, or outline, fonts. The following sections describe each of these font types in more detail. The *cyrillic* directory is optional and, as you might expect, it has Cyrillic fonts in it. Your system might not have the *PEX* directory which contains fonts for the PEX 3D graphics programming extension to X. These fonts are used only by PEX and aren't discussed further in this chapter.

Depending on your requirements and your interest in using a variety of fonts, you may find that you want to add more fonts--perhaps for internationalization support or for use with programs such as Netscape or the GIMP, which is described in more detail in Chapter 3, *A Selection of Useful X Clients*. You can add fonts to existing font directories or you can create new directories, so your *fonts* directory might not be exactly the same as the one shown above. The section "Adding and Removing Fonts" describes how to make new fonts available to X.

If You Just Want to Pick a Font

X font names can be so complicated that you might prefer not to deal with them at all. If you want to experiment with fonts or access fonts on remote machines, you'll have to spend some time familiarizing yourself with the conventions anyway. But if you just want to locate some fonts to use with *xterm* and other clients, you can use the predefined aliases for some of the constant-width fonts that should be available on most systems.

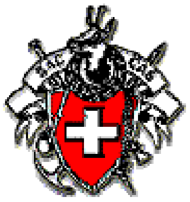


Figure 13-1: Aliases for some fixed-width fonts

Figure 13-1 lists the predefined aliases for some fixed-width fonts that should be appropriate for most of the standard clients, including *xterm*. To give you an idea of the range of sizes, each alias is written in the font it identifies.

Illustration department, I need help with this figure. It should look like Ellen's Figure 4-1 in the draft to this book which she sent me. There it's referenced as "lx-fon-f-xterm". The following figure, Swiss Alpine Club's logo, is just a place holder. If it should get printed by mistake you would my wife a joy!

These aliases refer to the dimensions in pixels of each character in the font. For example, "10x20" is the alias for a font with characters 10 pixels wide by 20 pixels high. Note, however, that an alias can be virtually any character string.

The default font for many applications, including *xterm*, is a 6 x 13 pixel font that has two aliases: `fixed` and `6x13`.[*] Many users consider this font too small. If you have enough screen space, you might want to use the 10 x 20 font for *xterm* windows:

```
xterm -fn 10x20 &
```

You can make this font the default for *xterm* by specifying it as the value for the `font` resource variable in a *.Xresources* or other resource file:

```
XTerm*font: 10x20
```

Table 13-1 shows the correspondence between the aliases for some of the fixed-width fonts and their full font names. Note that the 6 x 13 font has the additional alias `fixed`, which is the default font for *xterm* windows. (Twelve-point Helvetica bold roman, a proportional font often used as the default label font, has the alias `variable`. *fvwm*, for example, uses this font for the text that appears in menus, icons, and titlebars.)

Table 13-1: Fixed Font Aliases and Font Names

Alias	Filename
5x7	-misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
5x8	-misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-1
6x9	-misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
6x10	-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
6x12	-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
6x13	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
fixed	-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
6x13bold	-misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-1
7x13	-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1

[*] XFree86 4.0 redefines this font to a Unicode version that includes support for many international characters and technical symbols; the new version is compatible with the old `fixed` font, so you shouldn't notice the change other than that many more characters are available. See the section "Using International Fonts" for more information on Unicode.

7x13bold	-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1
7x14	-misc-fixed-medium-r-normal--14-130-75-75-c-70-iso8859-1
7x14bold	-misc-fixed-bold-r-normal--14-130-75-75-c-70-iso8859-1
8x13	-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1
8x13bold	-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1
8x16	-sony-fixed-medium-r-normal--16-120-100-100-c-80-iso8859-1
9x15	-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso8859-1
9x15bold	-misc-fixed-bold-r-normal--15-140-75-75-c-90-iso8859-1
10x20	-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1
12x24	-sony-fixed-medium-r-normal--24-170-100-100-c-120-iso8859-1

Types of Fonts

The fonts available for X come in two major formats: *bitmap* and *scalable* (also known as *outline*). Bitmap fonts define each character as a grid of pixels; the appropriate pixels on the screen are turned on to display a character. Scalable fonts describe the characters by representing the lines and curves mathematically. Bitmap fonts require a file for each point size of the font, representing the characters at that point size, while scalable fonts only need one file for the entire range of sizes. When you request a scalable font at a particular point size, the information in the file is used to scale the displayed characters to that size.

In understanding the difference between a bitmap font and a scaled font, it helps to know that font technology makes a distinction between a *character* and a *glyph*. A character is a logical unit, while a glyph is a particular representation of the character. In fact, a glyph doesn't have to represent a single character--a ligature such as Æ is a glyph that incorporates the two characters A and E. On the other hand, a lowercase italic *a* and an uppercase roman A are both glyphs representing a single character. Thus one way to look at bitmap versus scaled fonts is that a bitmap font has multiple glyphs for each character, while a scaled font has one glyph that is scaled to size on demand.

In early releases of X, only bitmap fonts were provided. However, there are drawbacks to the use of bitmap fonts, not the least of which is that having multiple files for basically the same typeface wastes disk space and makes managing the fonts more difficult. Also, because of differences in monitor resolution, a font of a particular point size might actually appear larger or smaller, depending on the monitor. The standard X bitmap fonts are stored on your system in *portable compiled font* format; the files have the extension *.pcf*.

Current versions of X also include scalable fonts. The standard distribution of XFree86 includes *Speedo* and *Type1* font directories, both of which contain scalable fonts. The *Speedo* font directory has Charter and Courier scalable fonts donated by Bitstream, Inc. The *Type1* directory has several Adobe Type1 PostScript fonts, and it also has Type1 versions of the Bitstream fonts. You may also have, or want to consider installing, a collection of Type1 fonts from URW. These are high-quality freely available versions of the 35 standard PostScript fonts (as opposed to the few that are in the *Type1* directory). The Speedo fonts have an extension of *.spd*, while the Type1 fonts have an extension of either *.pfa* or *.pfb* and the URW fonts have *.pfb*.

Finally, you may want to use TrueType fonts on your Linux system, particularly if you are running Linux on a PC or a Macintosh, which already has TrueType fonts installed. Like the Speedo and Type1 fonts, TrueType fonts are scalable; they have the extension *.ttf*. TrueType fonts were specifically designed to look good on computer screens. XFree86 4.0 now includes TrueType support and even some of the current Linux distributions that still include XFree86 3.x also provide ways of using TrueType fonts. See the section “Serving TrueType Fonts” for more information.

Specifying Fonts

Most X clients let you override the default font and specify one that you prefer. As usual when setting resources, you can change the font for a single invocation of the client by setting it on the command line, or you can set it in a resource file so it takes effect each time you run the client. While it’s possible to set the font without really understanding how font names are specified and what the parts of the name are, it’s more efficient and ultimately more effective to know what you are doing.

The Font Name

The parts of the font name are:

```
fndry-fmly-slant-swid-adstyle-pxlsz-ptsz-resx-resy-spc-avgwdth-reg-encod
```

Figure 13-2 shows a full font name, tagged to show the components.

Illustration department, I need help with this figure. It should look like Ellen’s Figure 4-2 in the draft to this book which she sent me. There it’s referenced as “lx-fon-f-namecomp“. However, an important change: “character set” which points to “iso8859-1” should be changed to “registry” pointing to “iso” and “encoding” pointing to “8859-1”.

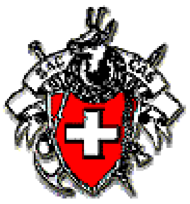


Figure 13-2: Font name components

While the next few sections explain each of these parts in more detail, Table 13-2 offers a quick reference to the parts.

Table 13-2: Font name parts

Part	**Description**
`fndry`	The foundry that digitized the font
`fmly`	The typeface family that the font belongs to

`wght`	The weight of the font; i.e., the degree of blackness
`slant`	The slant of the font
`swid`	The width of the font proportional to its horizontal size (the setwidth)
`adstyle`	Additional style information; e.g., whether the font is serif or sans serif
`pxlsz`	Pixel size; 0 for a scalable font
`ptsz`	Point size, in tenths of a point; 0 for a scalable font
`resx`	Horizontal resolution; 0 for a scalable font
`resy`	Vertical resolution; 0 for a scalable font
`spc`	Spacing; possible values are `p` (proportional), `m` (monospaced), or `c` (character cell)
`avgwdth`	The arithmetic mean width of all glyphs in the font, in tenths of a pixel
`reg`	The registration authority that owns the font's encoding
`encod`	The name of the encoded character set

The X Window System font-naming conventions are intended to allow for complete specification of all of the characteristics of each font. The full font name uniquely identifies the font. Once you understand how the name is put together, you'll also realize that there are parts of the name that you need to specify and other parts that you can safely ignore most of the time.

The naming conventions are officially known as the X Logical Font Description Conventions; they are defined in an X Standards document of that name that can be found at *ftp://ftp.x.org/pub/*. The next few sections describe each of the pieces of a font name. After that, we'll look at how to use wildcards and aliases to make life easier.

Foundry and family

Font manufacturers are still referred to as foundries, from the days when type was cast from lead. The X font-naming convention specifies the foundry as the company that digitized or last modified the font, rather than its original creator.

The term *font* is often used somewhat ambiguously. Sometimes it is used to refer to a typeface family (such as Times Roman or Helvetica) that come in different sizes, weights, and orientations. At other times it refers to a particular set of character glyphs within a typeface family. In that sense, for example, Adobe Courier is one font and Bitstream Courier is another.

For the most part, X takes the latter approach. When you read documentation that says that X includes more than 500 fonts, this sounds either intimidating or impressive, depending on your mood. In fact, the X distribution includes only eight font families (Charter, Courier, Helvetica, Lucida, New Century Schoolbook, Symbol, Times, and the Clean family of fixed-width fonts), plus individual sizes and orientations of several miscellaneous fonts, and a number of special-purpose fonts. Thus, if you think of the X fonts as comprising several large font families rather than as hundreds of individual fonts, you can quickly reduce the clutter.

The foundry is important in cases such as the earlier Courier example, where there are fonts on the system from more than one foundry. For fonts that are only supplied by one foundry (e.g., the Lucida font family), the foundry is less significant. In particular, many of the commercial font families are available from more than one foundry; if you start accumulating lots of fonts, you may need to specify the foundry to get the font you want.

In general, the appearance of fonts from the same family provided by different foundries should be quite similar, since the font family defines the design of the typeface. However, there may be some small differences in the quality of some of the characters, and there may be more significant differences in the font metrics (the vertical and horizontal measurements of the characters). This is a concern for applications that use a bitmap font for a *WYSIWYG* ("what you see is what you get") screen display that must match the fonts in a particular laser printer or typesetter.

In addition, when you select a font family, one consideration is whether you want a proportional or a fixed-width font. In a proportionally spaced font, each character has a different width; an l, for example, takes up less space than an m. This makes proportional fonts look good on a printed page but makes them less appropriate for screen display in a terminal window (especially for program editing), since text doesn't line up properly unless all characters are the same width. You're most likely use the proportional fonts for labels or menu items, or in an application like Netscape, rather than in an application like *xterm* or an Emacs window.

With a fixed-width font, on the other hand, every character takes up the same amount of space on the page, regardless of how narrow or how wide the character is--look at the difference between this `l` and `m`, compared to the proportional ones.

Just to complicate matters, X recognizes two different types of fixed-width font: monospaced and character cell. A monospaced font is just a standard fixed-width font where every character occupies the same amount of space. A character-cell font is a monospaced font specifically designed for computer displays, in which each character is surrounded by an invisible cell. Thus spacing is related to the size of the cell, rather than to the character itself. For text within *xterm* and other terminal emulators, you're better off using character cell fonts. Although you can use monospaced fonts as the text font in *xterm* windows, you may notice that some "garbage" pixels are occasionally left on the screen. This effect doesn't happen with character-cell fonts because the characters are clearly contained (or divided) from one another.

Courier and Lucidatypewriter are two examples of monospaced fonts, while Sony and Schumacher are examples of character-cell fonts. The *spacing* field in the font name indicates which of these three types a font belongs to. Many of the character-cell fonts have simple aliases that correspond to their dimensions in pixels; we saw many of those aliases earlier in Figure 13-1.

Weight and slant

The characters in any particular font family can be given a radically different appearance by changing the *weight* or the *slant*, or both. Weight is the degree of blackness of the font (that is, how dark or light the characters appear), while slant describes the posture of the characters.

The most common weights are medium and bold. The most common slants are roman (upright), italic, or oblique. Both italic and oblique are slanted; however, italic versions of a font generally have had the character shape changed for a more pleasing effect when slanted, while oblique fonts are simply a slanted version of the upright font. In general, *serif* fonts (those with little decorations on the ends and corners of the characters) are slanted via italics, while *sans serif* fonts are oblique. (Whether a font is serif or sans serif comes under the category of additional style; see the section "Other Information in the Font Name.")

Figure 13-3 compares the medium and bold weights, and the roman and italic or oblique slants in the Charter and Helvetica font families.

Illustration department, I need help with this figure. It should look like Ellen's Figure 4-3 in the draft to this book which she sent me. There it's referenced as "lx-fon-f-samefonts".

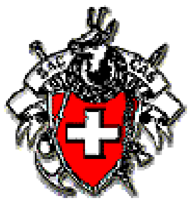


Figure 13-3: The same fonts in different weights and slants

X also includes the B&H lucidabright font, which has an in-between weight called *demibold*. Actually, weight names are somewhat arbitrary, since a demibold weight in one family may be almost as dark as a bold weight in another.

The font-naming convention also defines two counter-clockwise slants called *reverse italic* (ri) and *reverse oblique* (ro), as well as a catch-all called *other* (ot).

Pixel and point size

Font sizes are often given in a traditional printer's measure known as a *point*. A point is approximately one seventy-second of an inch. The characters in an X font are measured both in points and in pixels. Because there's only one file for each scalable font, there's no need for the font name to specify the point or pixel size. Thus, scalable font names have zeros in the columns for point and pixel size; when you specify the font to use, you fill in the size you want:

```
xfontsel -fn -adobe-utopia-medium-r-normal--0-120-0-0-p-0-iso8859-1 &
```

With a bitmap font, on the other hand, each size and orientation of the font must be stored in a separate file, and the point and pixel sizes are part of the font name. Bitmap fonts are optimized for each font size. Though they can be scaled, they are not intended to be, and scaled versions of bitmap fonts may have a jagged appearance. Most of the bitmap font families are provided in the six point sizes shown in Figure 13-4.

Illustration department, I need help with this figure. It should look like Ellen's Figure 4-4 in the draft to this book which she sent me. There it's referenced as "lx-fon-f-sixsizes".

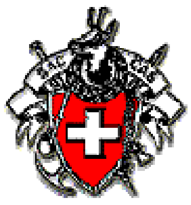


Figure 13-4: The same font in six different point sizes

An important feature of the scalable fonts is that they are device-independent and are true-to-size regardless of the screen. This is not true for bitmap fonts. Because of the different resolution of computer monitors, a bitmap font at a given point size might actually appear larger or smaller depending on the screen. The rest of this section discusses the factors that determine how large a font appears on your screen. This information might seem somewhat arcane, but if you're going to be dealing extensively with fonts you will need it. Everyone else can skip ahead to the next section.

Resolution is often spoken about in terms of the screen's dimensions in pixels. The *xdpyinfo* client gives you this information; your monitor documentation may also provide it. In addition to giving you the resolution in pixels, *xdpyinfo* tells you the screen's resolution in dots per inch. For example:

```
xdpyinfo
...
screen #0:
  dimensions:    1024x768 pixels (347x260 millimeters)
  resolution:    75x75 dots per inch
...
```

Most monitors on the market today have a resolution between 75 dots per inch (dpi) and 100 dots per inch. Accordingly, there are both 75-dpi and 100-dpi versions of most of the bitmap fonts in the standard distribution. These separate versions of each font are stored in different directories. By setting the font search path so that the directory with the version you want comes first, you'll get the correct versions without having to specify them in the font name (see the section "The Font Search Path" later in this chapter for more information.)

Note that the logical font-naming convention allows for different horizontal and vertical resolution values. In theory, this allows server manufacturers to support fonts that have been "tuned" to match individual monitor screen resolutions. However, the bitmap fonts shipped with the X distribution all have the same value for the horizontal and vertical resolutions. As suggested above, this resolution may not exactly match the actual resolution of any particular screen, resulting in characters that are not true to their nominal point size.

Specifying Scalable Fonts

If you’ve taken a look at the names of any of the scalable fonts, you may have noticed that the three fields relating to size (pixels, points, and average width), plus the two resolution fields, have zeroes in them. The following, for example, are the names of the fonts in the *Type1* directory (note that these are the names of the fonts, not the filenames):

```
-adobe-utopia-medium-r-normal--0-0-0-0-p-0-iso8859-1
-adobe-utopia-medium-i-normal--0-0-0-0-p-0-iso8859-1
-adobe-utopia-bold-r-normal--0-0-0-0-p-0-iso8859-1
-adobe-utopia-bold-i-normal--0-0-0-0-p-0-iso8859-1
-adobe-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
-adobe-courier-medium-i-normal--0-0-0-0-m-0-iso8859-1
-adobe-courier-bold-r-normal--0-0-0-0-m-0-iso8859-1
-adobe-courier-bold-i-normal--0-0-0-0-m-0-iso8859-1
-bitstream-charter-medium-r-normal--0-0-0-0-p-0-iso8859-1
-bitstream-charter-medium-i-normal--0-0-0-0-p-0-iso8859-1
-bitstream-charter-bold-r-normal--0-0-0-0-p-0-iso8859-1
-bitstream-charter-bold-i-normal--0-0-0-0-p-0-iso8859-1
-bitstream-courier-medium-r-normal--0-0-0-0-m-0-iso8859-1
-bitstream-courier-medium-i-normal--0-0-0-0-m-0-iso8859-1
-bitstream-courier-bold-r-normal--0-0-0-0-m-0-iso8859-1
-bitstream-courier-bold-i-normal--0-0-0-0-m-0-iso8859-1
```

You can specify one of these scalable outline fonts simply by inserting the point size you want, in tenths of a point, in the name at the time that you specify the font; for example:

```
xfd -fn -adobe-utopia-bold-r-normal--0-120-0-0-p-0-iso8859-1
&
```

Other information in the font name

The last few sections described the most important pieces of information in the font name--the ones you’re most likely to specify. The remaining fields are described briefly here:

Set width

> Set width describes the font’s proportionate width, according to the foundry. Typical set widths include: normal, condensed, semicondensed, narrow, double width. Most of the X fonts have the set width *normal*. A few fonts have the set width *semicondensed*.

Additional style

> This field is not represented in most font names. However, according to the logical font convention, the style of a font may be specified in the field between set width and pixels. Some of the possible styles are *i* (informal), *r* (roman), *serif*, and *sans* (sans serif). Currently, all the standard fonts have *sans* or an empty field. Note that the *r* for roman may also be used in the slant field.

Spacing

All standard X fonts are either: *p* (proportional); *m* (monospaced); or *c* (character cell). See the section “Foundry and family” for an explanation of the differences.

Average width

The mean width of all characters in the font, measured in tenths of a pixel. Two fonts (such as New Century Schoolbook and Times) with the same point size can have very different average character widths. This field can be useful if you want a font that is especially wide or especially narrow.

The Schumacher Clean family of fonts offers several fonts in the same point size but with different average widths.

Registry and encoding

These two fields together identify the organization or standard registering the character set and the actual character set (the encoding). Most fonts in the standard X distribution use the ISO 8859-1 character set, which represents the ISO Latin-1 character set and is specified as `iso8859-1`. In that case, `iso8859` is the registry and `1` is the encoding. The ISO Latin-1 character set is a superset of the standard ASCII character set that includes special characters used in European languages other than English.

Note, though, that the Adobe Symbol font contains the strings `adobe-fontspecific` in this position. This means that Adobe Systems defined the character set in this font, and that it is specific to this font. You can see from this example that the use of these fields is somewhat arbitrary.

For a complete technical description of font-naming conventions, see the X Consortium Standard, *X Logical Font Description Conventions*.

Using Wildcards

To simplify font specification, X supports the use of wildcards within font names. An asterisk (`*`) can be used to represent any part of the font-name string; a question mark (`?`) can be used to represent any single character. You can usually get the font you want by specifying the font family, the weight, the slant, and the point size, and using wildcards for the remaining fields. For example, to get Courier bold at 14 points, you can use the command-line option:

```
-fn '*courier-bold-r*140*'
```

That’s starting to seem a little more intuitive!

However, there are a few “gotchas:”

- Since the shell also recognizes the `*` and `?` wildcard characters, wildcarded font names used on the command line must be quoted, as in the example above. Inside a resource file, however, wildcards do not need to be quoted.
- Within a particular font directory, if the wildcarded font name matches more than one font, the X server uses the first one that matches. For example, if you don’t specify the weight, bold will be selected before medium because the names are sorted in simple alphabetical order.

- Across font directories, matches are resolved in font-path order--X uses the first matching font in the first directory that matches.
- The asterisk (*) wildcard expansion is resolved by a simple string comparison. So, for example, if you type:

  ```
  -fn '*courier-bold*r*140*'
  ```

  instead of:

  ```
  -fn '*courier-bold-r*140*'
  ```

  where the difference is the asterisk instead of the hyphen before the `r` in the slant field, the `r` also matches the `r` in the string `normal` in the set width field (which comes right after the slant). The result is that you will select all slants. Since `o` (oblique) comes before `r` (roman), and you always get the first font that matches, you'll end up with Courier oblique. The trick is to be sure to include at least one of the hyphens to set the `-r-` off as a separate field rather than as part of another string.
- Font names are case-insensitive. "Courier" is the same as "courier."

If you aren't sure whether your wildcarded name is specific enough, try using it as an argument to the *xlsfonts* command, described more completely in the section "Listing Available Fonts". If you get more than one font name as output, you might need to use a more specific name string to be sure of getting the font that you want:

```
xlsfonts -fn '*courier-bold-r*140*'
-adobe-courier-bold-r-normal--14-140-75-75-m-90-
iso8859-1
-adobe-courier-bold-r-normal--20-140-100-100-m-110-
iso8859-1
-adobe-courier-bold-r-normal--20-140-100-100-m-110-
iso8859-1
-cronyx-courier-bold-r-normal--20-140-100-100-m-120-
koi8-r
```

Table 13-3 summarizes the values you should always use to specify a font name (assuming only the standard fonts are loaded). Choose one element from each column. Don't forget to include the leading and trailing asterisks and the hyphen before the slant.

Table 13-3: Essential Elements of a Font Name

	Family		Weight		Slant		Point Size	

*	charter clean courier fixed gothic helvetica lucida lucidabright lucidatypewriter mincho new century schoolbook nil open look cursor open look glyph symbol terminal times	-	medium bold demibold	-	r (roman) l (italic) o (oblique) ri (reverse italic) ro (reverse oblique) ot (other)	*	80 (8 pt.) 100 (10 pt.) 120 (12 pt.) 140 (14 pt.) 180 (18 pt.) 240 (24 pt.)	*

The point sizes listed in the table correspond to the point sizes of the standard bitmap fonts in the *75dpi* and *100dpi* directories. If you are specifying one of the scalable outline fonts, you do not have to limit yourself to these sizes; pick any size you want. See "Selecting Fonts With xfontsel" for information about previewing fonts.

Using Aliases

For the bitmap fonts, you can simplify the task of abbreviating font names by aliasing them; that is, associating the fonts with alternative names of your choosing. You can edit or create a file called *fonts.alias*, in any directory in the font search path, to set aliases for existing fonts. The X server uses both the *fonts.dir*, described later in the section "The fonts.dir File", and *fonts.alias* files to locate fonts.

X provides a default *fonts.alias* file for each of the three bitmap font directories (*misc*, *75dpi*, and *100dpi*) and for the *cyrillic* directory. Take the time to look at the contents of each of these files, since many of the existing aliases may reference fonts you use and may be easier to type than even wildcarded font names.

However, the scalable fonts cannot be aliased because you must provide a point size when you specify them. If there are scalable fonts that you use regularly with certain applications, you'll probably want to specify them in a resource file as described in Chapter 12, *Setting Resources*.

You can add your own aliases to the *fonts.alias* file, change existing aliases, or even replace the entire file. However, it's safer to add entries to an existing *fonts.alias* file than to change what is there, especially if you're working in a multiuser environment where someone else might be using those aliases.

The *fonts.alias* file has a two-column format. The first column contains the aliases and the second column contains the actual font names; for example, here are a few entries from the *misc/fonts.alias* file:

```
fixed     -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
variable  -*-helvetica-bold-r-normal-*-*-120-*-*-*-*-iso8859-1
5x7       -misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
5x8       -misc-fixed-medium-r-normal--8-80-75-75-c-50-iso646.1991-irv
6x9       -misc-fixed-medium-r-normal--9-90-75-75-c-60-iso646.1991-irv
6x10      -misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
```

To specify an alias that contains spaces, enclose the alias in double quotes. If you want double quotes (`"`) or other special characters as part of an alias name, precede each special symbol with a backslash (`\`).

After you create or edit a *fonts.alias* file, you need to make the server aware of the change by resetting the font path with the *xset* command. See the section "Modifying the Font Search Path" for details.

When you use an alias to specify a font in a command line, the server searches for the font associated with that alias in every directory in the font path. Therefore, a *fonts.alias* file in one directory can set aliases for fonts in other directories as well. You might choose to create a single aliases file in one directory of the font path to set aliases for the most commonly used fonts in all the directories. The following shows three sample entries that could be added to an existing *fonts.alias* file or used to create a new one:

```
cour12     -adobe-courier-medium-r-normal--12-120-75-75-m-70-iso8859-1
cour14     -adobe-courier-medium-r-normal--14-140-75-75-m-90-iso8859-1
cour18     -adobe-courier-medium-r-normal--18-180-75-75-m-110-iso8859-1
```

As the names of the aliases suggest, these sample entries provide aliases for three Adobe Courier fonts of different point sizes. You can include wildcards in the right-hand column, which specifies the real font names. For instance, the alias file entries above can also be written as follows:

```
cour12     -adobe-courier-medium-r-*-120*
cour14     -adobe-courier-medium-r-*-140*
cour18     -adobe-courier-medium-r-*-180*
```

If you want to set up aliases for all the fonts in a directory, a quick way to do it is to assign each font its base filename as an alias. For example, the font `-adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1` is stored in the file *courBO18.pcf.gz* in the directory */usr/X11R6/lib/X11/fonts/75dpi*. So let's create the alias `courBO18` for that font (and equivalent aliases for the other fonts) in the *75dpi* directory. The *fonts.dir* file, described later in the section "The fonts.dir File", contains a line like the following for each font in the directory:

```
courBO18.pcf.gz -adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1
```

Since both the *fonts.dir* and *fonts.alias* files have a two-column format with the font name in the second column, it's easy to turn a filename entry from *fonts.dir* into an entry for *fonts.alias* by simply removing the extension (i.e.,*.pcf.gz* in the example above) and adding the result to *fonts.alias*. In this example, the resulting alias looks like this:

```
courBO18 -adobe-courier-bold-o-normal--18-180-75-75-m-110-iso8859-1
```

Once the server has been made aware of the new alias (see the next section), you can start using it on the command line. For example:

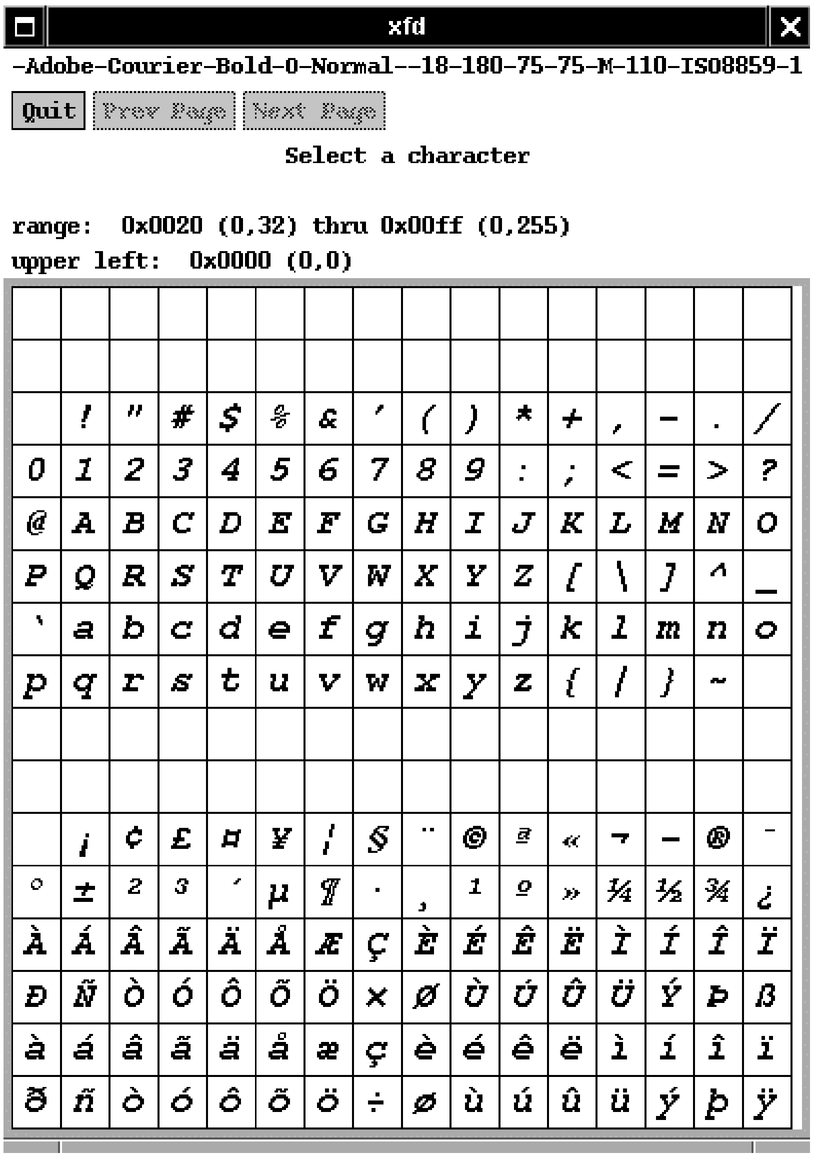
```
xterm -fn courBO18 &
```

starts a new *xterm* with the 18-point Courier oblique font--an interesting font choice for those who left their glasses at home.

Or you might want to display the characters in the font:

```
xfd -fn courBO18 &
```

as seen in Figure 13-5. *xfd* is described further in the section “The Font Displayer: xfd”.



```
-Adobe-Courier-Bold-O-Normal--18-180-75-75-M-110-ISO8859-1
Select a character
range:  0x0020 (0,32) thru 0x00ff (0,255)
upper left:  0x0000 (0,0)
```

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Figure 13-5: Characters in the 18-point Courier oblique font

If you edit the *fonts.alias* file in the *misc* directory, be careful to preserve at least the `fixed` and `variable` aliases because they are widely used as system defaults. Better yet, keep all the default entries and simply append any new ones.

Making the X server aware of aliases

Simply creating or updating an alias file does not mean that the X server automatically recognizes the aliases in question. If you want to use the aliases in the current X session, you must make the server aware of the newly created or edited alias files by “rehashing” the font path with the *xset* command:

```
xset fp rehash
```

on the command line. The *xset* font path option *fp* with the `rehash` argument causes the server to reread the *fonts.dir* and *fonts.alias* files in the current font path. You need to do this every time you edit an alias file. If you are running the X Font Server, you need to first restart the font server (as root) and then rehash the font path:

```
[root]# /etc/rc.d/init.d/xfs restart
xset fp rehash
```

Selecting Fonts With xfontsel

The *xfontsel* client provides a font previewer window in which you can look at and select fonts by varying each of the components of the name. *xfontsel* lets you dynamically change the font displayed in the window; this is particularly useful if you want to pick a display font, and don’t have a clear idea which font you want.

Once you’ve found the font you want, you can make the name of that font the PRIMARY text selection by clicking on the window’s `select` button. You can then paste the font name into another window: onto a command line, into a resource file, etc. Making a font name the PRIMARY selection also enables you to choose that font from the *xterm* `VT Fonts` menu. (See the section “Changing Fonts in an xterm Window”.)

To run *xfontsel*, enter the command in an *xterm* window:

```
xfontsel &
```

Figure 13-6 shows the initial *xfontsel* window.

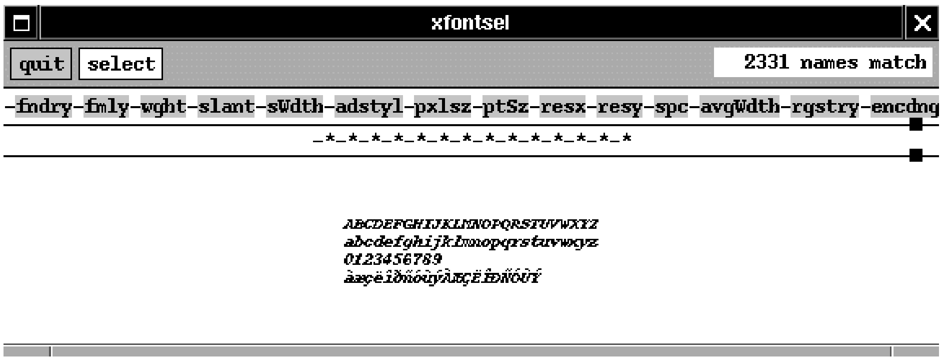

xfontsel
quit
select
2331 names match
-fndry-fmly-wght-slant-sWdth-adstyl-pxlsz-ptSz-resx-resy-spc-avgWdth-rgstry-encdng
-*-*-*-*-*-*-*-*-*-*-*-*-*-*
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

Figure 13-6: Initial xfontsel window

There are two command buttons on the first line of the *xfontsel* window: `quit` and `select`. As we've seen, clicking on `select` with the first pointer button makes the font displayed in the window the PRIMARY text selection; `quit` causes the application to exit.

Below the command buttons is, in effect, a generic font name, divided into 14 fields corresponding to the 14 parts of a standard font name. Each of the fields in the *xfontsel* window is actually the handle to a menu which lets you specify this part of the font name.

To get a clearer idea of how this works, move the pointer onto the generic font name--specifically onto the first field, `fndry`, representing the first part of a font name, the foundry. The field title is now highlighted by a box. When you click on the first pointer button, you get a menu of foundry options, as shown in Figure 13-7.

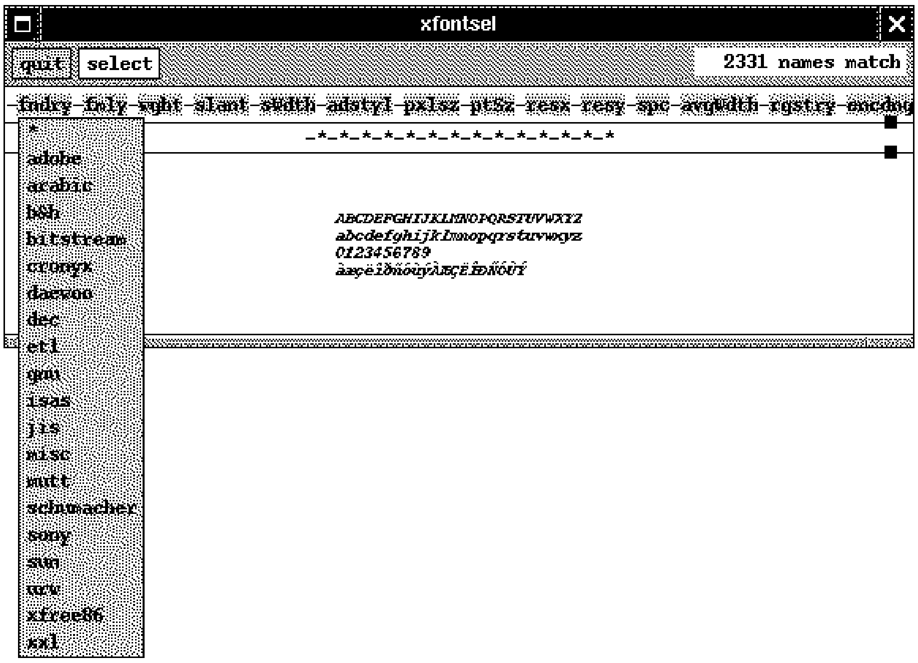


Figure 13-7: xfontsel window with foundry menu displayed

The first choice on each menu is the asterisk (`*`) wildcard character, followed by the remaining choices--only the options that are available given earlier choices are highlighted, while the rest are grayed out. For example, if you choose `b&h` from the foundry menu and then click on `fmly`, your choices are restricted to the three `lucida` fonts.

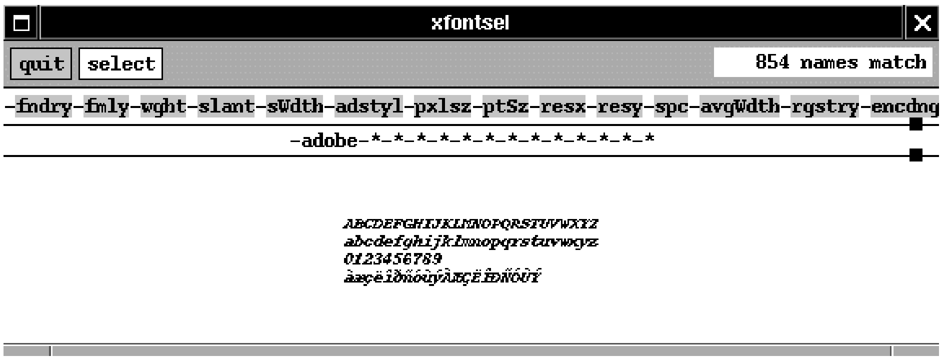
The font name being selected is built on the third line. When you run *xfontsel*, all the fields on the third line initially contain wildcard characters because no menu selections have been made. The number of fonts matched by the font name is displayed in the upper-right corner of the window. The number of fonts initially matched depends on the number of fonts with this naming convention available on your system, in this case 2331 fonts. Since this line of wildcards matches *every* 14-part font name, *xfontsel* chooses the first font in the font path to display.

When you select a font name component, the equivalent * is replaced with your selection in the font name on the third line, and the bottom portion of the window displays the first font that matches this name. For example, if we select adobe from the fndry menu, the *xfontsel* window changes to look like Figure 13-8 and the font name becomes:

```
-adobe-*-*-*-*-*-*-*-*-*-*-*-*-*
```

In this case, the first font to match is a 10-point bold Oblique Courier font, with the filename *courBO10.pcf*. The actual font name is:

```
-adobe-courier-bold-o-normal--10-100-75-75-m-60-iso8859-1
```



xfontsel

quit select 854 names match

-fndry-fmly-wght-slant-sWdth-adstyl-pxlsz-ptSz-resx-resy-spc-avgWdth-rgstry-encdng

-adobe-*-*-*-*-*-*-*-*-*-*-*-*-*

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
àæçèìðñòùýÅÆÇÈÌÐÑÒÙÝ

Figure 13-8: xfontsel after choosing adobe from the foundry menu

Once you make a selection from one menu, the number of possible fonts matched by the name changes. (Notice the line 854 fonts match in the upper-right corner of the window.) Choosing one font name component also eliminates certain choices on other menus and the unavailable choices are grayed out.

In order to display any given font, you'll probably have to make selections from several of the menus. You'll probably want to at least select the font family, weight, slant, point size. Thus, you would make selections from the fmly, wght, slant, and ptSz menus. Basically, you need to specify just enough information so that only one font matches your specification.

You can also start with a partial selection, to limit the number of choices, by running *xfontsel* with the *–pattern* option and a wildcarded font name. For example, if you type:

```
xfontsel -pattern '*courier-bold-o-*'
```

you start out with the specified pattern in the filename template part of the *xfontsel* display. You can then select from the ptSz menu to view Courier bold oblique at various point sizes until you find a size you like.

Note that if the pattern you specify to *xfontsel* matches more than one font, the one that is displayed (the first match found) is the one that the server will use if you select that pattern. You can always rely on *xfontsel* to show you the actual font that will be chosen, given any wildcard specification.

The default is for *xfontsel* to display the alphabetic characters, the digits 0 through 9, and a range of international characters, or the equivalent characters (i.e., those with the same key codes) in a non-alphabetic font. However, if there are particular characters you want to see, you can select the text to be displayed by including the *–sample* option:

```
xfontsel -sample 'abcABC!@#$%^&*'
```

One thing to note about *xfontsel* is that it only displays fonts from directories that are in the font search path. See the section “Modifying the Font Search Path” to add or delete directories containing fonts. For example, if the list of choices for foundry doesn’t include `cronyx`, which is the foundry for the Cyrillic fonts, then Cyrillic fonts aren’t in your font path. Issuing the command:

```
xset fp+ /usr/X11R6/lib/X11/fonts/cyrillic
```

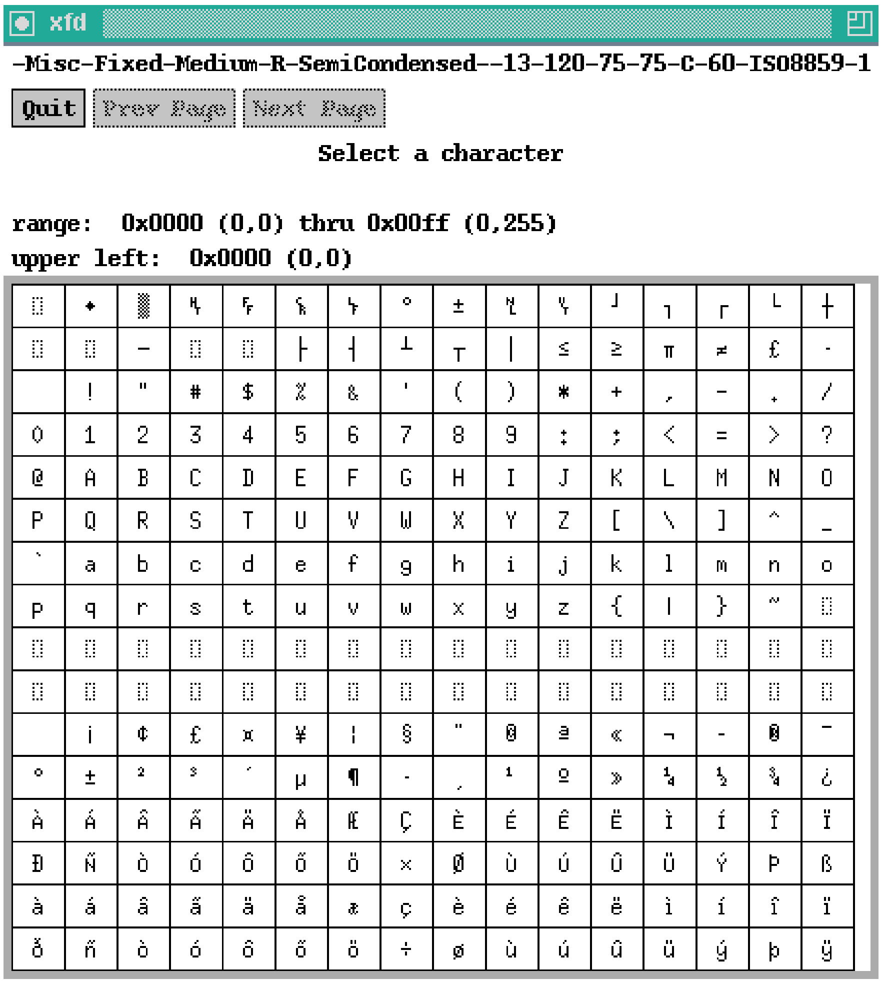
appends the *cyrillic* directory to the font path for this session and running *xfontsel* now shows `cronyx`. If you did much work with Cyrillic fonts, you should add it to your font path in *XF86Config*. For more information on *xset*, see the section “Modifying the Font Search Path”.

The Font Displayer: xfd

You can display the characters in a font using the *xfd* (X font displayer) client, using the *–fn* option to specify the name of the font to display. For example, to display the default system font `fixed`, enter:

```
xfd -fn fixed &
```

The *xfd* window displays the specified font as shown in Figure 13-9. The font name is displayed across the top of the window. (This is the actual font name, not the alias we specified on the command line.) Three command buttons appear in the upper-left corner of the window below the font name. If the font being displayed doesn’t fit within a single *xfd* screen, `Prev Page` and `Next Page` allow you to scroll through multiple screens. The `Quit` button causes the application to exit; you can also exit by typing `q` or `Q` while input is focused on the *xfd* window.



-Misc-Fixed-Medium-R-SemiCondensed--13-120-75-75-C-60-ISO8859-1

Select a character

range: 0x0000 (0,0) thru 0x00ff (0,255)
upper left: 0x0000 (0,0)

⬚	◆	▒	␉	␌	␍	␊	°	±	␤	␋	┘	┐	┌	└	┼
⬚	⬚	─	⬚	⬚	├	┤	┴	┬	│	≤	≥	π	≠	£	·
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⬚
⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚
⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚	⬚
	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Figure 13-9: Fixed font, 6x13 pixels

In addition to displaying a font, *xfd* lets you display certain information about the individual characters. But before we examine these capabilities, let's take a closer look at the way the characters in a font are identified and how the *xfd* window makes use of this information.

Within a font, each character is considered to be numbered. The *xfd* client displays a font's characters in a grid. By default, the first character appears in the upper-left position; this is character number 0. The two text lines above the grid identify the upper-left character and the range of characters in the window by character numbers, both in hexadecimal and in decimal notation (the decimal is in parentheses following the hex character number).

You can specify a character other than character number 0 to be in the first position in the window using the *–start* option. For example, if you enter this command line:

```
xfd -start 15 -fn fixed &
```

the *xfd* window begins with character number 15 (the 16th character) and continues to the last character. To see the earlier characters, press the `Prev Page` key.

Notice the instruction `Select a character` below the command buttons. To display information about a particular character, click any pointer button within the character's grid square. Statistics about the character's number, width, left bearing, right bearing, ascent, and descent are displayed where the line `Select a character` previously appeared. *xfd*'s *–box* parameter places a rectangle around each character, showing where a program might display a background color. The terms width, left bearing, right bearing, ascent, and descent are various metrics that describe the character .

The *xfd* client is most useful when you have an idea what font you might want to display. If you don't have a particular font in mind or would like to survey the possibilities, the *xfontsel* client allows you to preview multiple fonts interactively, as we've seen.

The Font Search Path

The font search path defines the order in which the X server searches font directories when a client requests a font to be displayed. The standard font path looks like this (depending on your Linux distribution and X version):

```
FontPath    "/usr/X11R6/lib/X11/fonts/misc"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
```

The standard path has the bitmap font directories first (*misc*, *75dpi*, and *100dpi*), then the scalable font directories (*Speedo* and *Type1*).

If you are using a font server (see the section "Font Servers"), your font path may include an entry for the server in addition to the other entries. In the following example, the font server entry is placed at the end of the path, so it will be searched last:

```
FontPath    "/usr/X11R6/lib/X11/fonts/misc/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/URW/"
FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
FontPath    "tcp/kansas:7100"
```

The examples above show the traditional way to set the font search path. Nowadays, it's more common to offload the serving of fonts entirely from the X server by running a local font server. To do that, you include only an entry for the server in *XF86Config*, and define the font path itself in the font server configuration file */etc/X11/fs/config*.

The font-path settings for systems running a font server are discussed in the section "Font Servers".

Finally, if your monitor and video card support the higher resolution, you might want to switch the order of the *75dpi* and *100dpi* directories, so the *100dpi* directory is searched first.

Modifying the Font Search Path

When X is installed, a default font path is set up in the file */etc/XF86Config* (or */etc/X11/XF86Config*). You can change the font path for a single X session using the *xset* client with the *fp* (font path) option, or you can change the font path permanently by modifying *XF86Config*. Some common reasons for changing the font path are to:

- Change the order in which directories are searched
- Add new font directories or remove directories you no longer use
- Add the `:unscaled` option to the bitmap directories so a scalable version of a font will be used rather than a scaled bitmap font
- Add a font server to the search path
- Rearrange the order of the directories in the font search path
- Add directories to or subtract directories from the font path
- Completely replace the font path

Let's assume that you've added the `:unscaled` option to use scalable fonts instead of a scaled bitmap font. In that case, your font path will look something like this:

```
FontPath    "/usr/X11R6/lib/X11/fonts/misc/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
FontPath    "/usr/X11R6/lib/X11/fonts/URW/"
FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
```

Whenever you change the font path, or add or remove fonts from your system, you need to let the X server know about the changes. For example, say you add fonts to a directory called *newfonts*, a subdirectory of your home directory.[*] The following two sections tell your X server to use these new fonts. Using the *xset* command is a temporary solution that makes the change for the current session, while modifying *XF86Config* makes the change permanent. In fact, you can use both techniques together. If you modify *XF86Config*, you need to restart the X server for the change to take effect. To avoid doing that while you are working, you can also run the *xset* command and use the change for the current session.

[*] After you add fonts to a directory, but before you run *xset* or modify *XF86Config*, you must first run the command:

```
mkfontdir directory_name
```

The section "The fonts.dir File" explains more about that.

Using the xset command

To access the new fonts, add the directory to the font path using *xset*. To append a directory (or list of directories) to the end of the font path, use the `fp+` option. The following command appends the *~/newfonts* directory to the path:

```
xset fp+ ~/newfonts
```

We can verify that the new path is in effect by running *xset* with the *q* option to query the settings:[*]

```
xset q
  ... lots of output
Font Path:
  /usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/Type1/,/usr/X11R6/
lib/X11/fonts/Speedo/,/usr/X11R6/lib/X11/fonts/75dpi/,/usr/X11R6/lib/X11/fon
ts/100dpi/,~/newfonts
  ...
```

(The path simply wraps on your screen, depending on the size of your *xterm* window and the actual font path on your system.)

When you add directories to the font search path, the location of the plus sign is significant. The command:

```
xset +fp ~/newfonts
```

prepends *~/newfonts* to the beginning of the existing path.

You can remove a directory from the font path using either the *–fp* or *fp–* option. In this case, the location of the minus sign is not significant. Thus, to remove *~/newfonts* from the font path, you can enter either of the following:

```
xset -fp ~/newfonts
xset fp- ~/newfonts
```

You can add or subtract multiple directories from the font path with a single invocation of *xset*. After the relevant *fp* option, the argument should be a comma-separated list of directories, as in the command:

```
xset fp+ ~/newfonts,/usr/X11R6/lib/X11/fonts/new
```

This command adds the directories *~/newfonts* and */usr/X11R6/lib/X11/fonts/new* to the end of the current font path.

To completely replace the font path, use *fp=* followed by a comma-separated list of directories:

```
xset fp= fontdir1,fontdir2,fontdir3,...
```

Note that a space must follow the equal sign (`=`).

You also use `xset fp=` to change the order of directories in the current path. For example, to put the *100dpi* directory before the *75dpi* directory, enter:

```
xset fp= /usr/X11R6/lib/X11/fonts/misc,/usr/X11R6/lib/X11/fonts/Speedo,\
```

[*] Not only does *xset –q* query the font path, it also queries many user preferences as described in Chapter 12, *Setting Resources*.

```
         /usr/X11R6/lib/X11/fonts/100dpi,/usr/X11R6/lib/X11/fonts/75dpi
```

You can restore the default font path at any time by entering:

```
xset fp default
```

Note that if you want to access fonts via a remote (or even a local) font server (*xfs*) program, you must add the font server to the path. See “Running the X Font Server” later in this chapter for more information.

Modifying the XF86Config file

XFree86’s configuration file, *XF86Config*, is usually found in either */etc/X11/* or */usr/X11R6/lib/X11/*, depending on your Linux and XFree86 versions. In some cases, if you are running XFree86 4.0, it may be named *XF86Config-4*. The `FontPath` entries tell *X* where to find the fonts it uses to render text on your display.

Adding the line

```
FontPath    "/home/ellen/newfonts/"
```

to the existing lines in your *XF86Config* configuration file informs the X server to also look in the */home/ellen/newfonts* directory for its fonts. Notice that the directory’s absolute path is used, not its relative path as shown for *xset*.

Before this change takes effect, you must restart your X server. If you’re using *xdm* or one of its derivatives, simply log out and log in again. You can run the command:

```
xset q
```

to verify that the font path has been updated.

The fonts.dir File

In addition to font files, each font directory contains a file called *fonts.dir*. The *fonts.dir* files serve, in effect, as databases for the X server. When the X server searches the directories in the default font path, it uses the *fonts.dir* files to locate the fonts it needs.

Each *fonts.dir* file contains a list of all the font files in that directory with their associated font names. The first line in *fonts.dir* lists the number of entries in the file (i.e., the number of fonts in the directory). After that, each line lists the filename of a font file in the directory, followed by a space and the name of the font associated with the file.

Example 4-1 shows a portion of *fonts.dir* from the */usr/X11R6/lib/X11/fonts/100dpi* directory. The first line in the file indicates that the directory contains 200 fonts. The full file is too long to show here; the example shows the fonts in the Courier family and a few sizes of fonts in the Charter family.

Example 4-1: *Subsection of fonts.dir file in /usr/X11R6/lib/X11/fonts/100dpi*

```
200
 ...
courBO08.snf -adobe-courier-bold-o-normal--11-80-100-100-m-60-iso8859-1
courBO10.snf -adobe-courier-bold-o-normal--14-100-100-100-m-90-iso8859-1
courBO12.snf -adobe-courier-bold-o-normal--17-120-100-100-m-100-iso8859-1
courBO14.snf -adobe-courier-bold-o-normal--20-140-100-100-m-110-iso8859-1
courBO18.snf -adobe-courier-bold-o-normal--25-180-100-100-m-150-iso8859-1
courBO24.snf -adobe-courier-bold-o-normal--34-240-100-100-m-200-iso8859-1
courB08.snf -adobe-courier-bold-r-normal--11-80-100-100-m-60-iso8859-1
```

```
courB10.snf -adobe-courier-bold-r-normal--14-100-100-100-m-90-iso8859-1
courB12.snf -adobe-courier-bold-r-normal--17-120-100-100-m-100-iso8859-1
courB14.snf -adobe-courier-bold-r-normal--20-140-100-100-m-110-iso8859-1
courB18.snf -adobe-courier-bold-r-normal--25-180-100-100-m-150-iso8859-1
courB24.snf -adobe-courier-bold-r-normal--34-240-100-100-m-200-iso8859-1
courO08.snf -adobe-courier-medium-o-normal--11-80-100-100-m-60-iso8859-1
courO10.snf -adobe-courier-medium-o-normal--14-100-100-100-m-90-iso8859-1
courO12.snf -adobe-courier-medium-o-normal--17-120-100-100-m-100-iso8859-1
courO14.snf -adobe-courier-medium-o-normal--20-140-100-100-m-110-iso8859-1
courO18.snf -adobe-courier-medium-o-normal--25-180-100-100-m-150-iso8859-1
courO24.snf -adobe-courier-medium-o-normal--34-240-100-100-m-200-iso8859-1
courR08.snf -adobe-courier-medium-r-normal--11-80-100-100-m-60-iso8859-1
courR10.snf -adobe-courier-medium-r-normal--14-100-100-100-m-90-iso8859-1
courR12.snf -adobe-courier-medium-r-normal--17-120-100-100-m-100-iso8859-1
courR14.snf -adobe-courier-medium-r-normal--20-140-100-100-m-110-iso8859-1
courR18.snf -adobe-courier-medium-r-normal--25-180-100-100-m-150-iso8859-1
courR24.snf -adobe-courier-medium-r-normal--34-240-100-100-m-200-iso8859-1
 ...
charBI08.snf -bitstream-charter-bold-i-normal--11-80-100-100-p-68-iso8859-1
charBI10.snf -bitstream-charter-bold-i-normal--14-100-100-100-p-86-iso8859-1
charBI12.snf -bitstream-charter-bold-i-normal--17-120-100-100-p-105-iso8859-1
charBI14.snf -bitstream-charter-bold-i-normal--19-140-100-100-p-117-iso8859-1
charBI18.snf -bitstream-charter-bold-i-normal--25-180-100-100-p-154-iso8859-1
charBI24.snf -bitstream-charter-bold-i-normal--33-240-100-100-p-203-iso8859-1
 ...
```

We mentioned earlier that you need to run the *mkfontdir* client whenever you make changes to the fonts on the system. The *fonts.dir* files are created by *mkfontdir* when X is installed or whenever you add new fonts or remove fonts. In the case of bitmap fonts, *mkfontdir* reads the font files in each directory in the font path, extracts the font names, and creates a *fonts.dir* file in the directory. The scalable outline fonts in the *Speedo* and *Type1* directories require an intermediary file called *fonts.scale*, which *mkfontdir* converts to a *fonts.dir* file. The standard distribution includes *fonts.scale* files in the font directories.

If *fonts.dir* files are present on your system, you probably won't have to deal with them or with *mkfontdir* at all. If the files are missing, or whenever you add or remove any fonts, you need to run *mkfontdir*. Running the command is straightforward--just type the command, including a directory name if you aren't already in the directory. For example:

```
mkfontdir /usr/X11R6/lib/X11/fonts/newfonts
```

This command creates a *fonts.dir* in the *newfonts* directory.

Font Servers

Up until now we've been talking about how to use fonts that are present on your X server. However, you aren't limited to your own fonts--X provides support for a *font server* that lets you provide access to your fonts to users on other systems. You might also want to run a font server on your system even if you aren't offering remote access, to offload font management from the X server. This is especially interesting if you want to use TrueType fonts used by Microsoft Windows or Macintosh systems.

The X Font Server: xfs

The font server program (*xfs*) acts as intermediary between your X server and the fonts resident on the system where the font server is running. You can add font servers running on remote machines to the font path on your local host and then access the remote fonts.*

In addition to using remote font servers, you can run a font server on your local system and add it to your font path, though the advantages of this are more subtle. Primarily, having the local font server in your path enables you to use certain font server clients (described later in this chapter) with local fonts. (You might, of course, want to run a font server locally to make your fonts available to users on other systems, as well as for your own use.)

Once you've added a font server to your path, you can request a font in the normal manner (e.g., on the command line, in a resource file, etc.). Font access is transparent--both local and remote directories are automatically searched. In the next sections, we'll take a quick look at the configuration file, how to run the font server program, and we'll look in more detail at how to add servers to the font path. Finally, we'll consider some of the clients that are available for the font server.

The font server configuration file

The operation of the font server program is controlled by a configuration file called *config*, in the directory */etc/X11/fs*. A sample configuration file comes with your distribution and should be sufficient for many environments. The font server *config* file is fairly straightforward. Here is a distribution-dependent sample file (the `catalogue` line is broken to fit on the page):

```
# font server configuration file
# $XConsortium: config.cpp,v 1.7 91/08/22 11:39:59 rws Exp $

clone-self = on
use-syslog = off
catalogue = /usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/Speedo/,\
/usr/X11R6/lib/X11/fonts/75dpi/,/usr/X11R6/lib/X11/fonts/100dpi/
error-file = /var/log/fs-errors
# in decipoints
default-point-size = 120
default-resolutions = 75,75,100,100
```

This file sets the following variables:

`Clone-self`
: The default value of `on` specifies that the font server should spawn another copy of itself to handle more than 10 client connections (the default maximum).

`use-syslog`
: The default value of `off` specifies that error messages are not to be logged to *syslog*.

* Technically, if the font server is not already running on a host whose fonts you want to use, you can run it yourself--presuming you have superuser privileges on that host.

`catalogue`
: A list of the font directories available from the server. By default, `catalogue` is the standard font path.

`error-file`
: An error log file to be used as an alternative to *syslog*.

`default-point-size`
: The default point size, specified in tenths of a point; thus, the default is 12 points.

`default-resolutions`
: The default horizontal and vertical resolutions, often 75x75 and 100x100.

The *config* file defaults should be reasonable for most environments, although you might need to change the value of the `catalogue` line to match your font path or update it if you add or remove any font directories..

Running the X font server

Though you can use fonts from any system that is running an X font server and that you have been authorized to access, in this section we describe how to install an X font server on your system for your own use.

You can check that the font server is running on your system by searching the process list to see if the *xfs* process is running:

```
ps aux | grep xfs | grep –v grep*
```

If the font server is running, you'll see output containing a line similar to this:

```
root   471  0.4  4.1  3712 2676 ? S   17:00   0:00 /usr/X11R6/bin/xfs
```

which corresponds to the *xfs* process. If the font server is running, you can skip ahead to the next section (see "Adding a server to the font path").

If *xfs* is not running on your system start it yourself (as root) with:

```
[root]# xfs &
```

This keeps the font server running until you reboot your system. A more permanent solution is to add the proper startup file in */etc/init.d/*[*] and links in */etc/init.d/rc3.d/* and possibly */etc/init.d/rc5.d*. Example 4-2 shows an example */etc/init.d/xfs* startup file.

Example 4-2: xfs startup file

```
#! /bin/bash
#
# Author: Werner Klauser
#
# /etc/init.d/xfs
#
```

[*] Adding "*| grep –v grep*" when searching the process list with *grep* avoids finding your own *grep* command.

[*] The location of the scripts for controlling the system is very distribution-dependent. The Linux Standard Base (LSB) specification puts them in */etc/init.d/* but you might find them in */etc/rc.d/init.d/* or even */sbin/init.d/*.

```
XFS=/usr/X11R6/bin/xfs

case "$1" in
    start)
        echo "Starting X font server "
        startproc $XFS
        ;;
    stop)
        echo "Shutting down X font server "
        killproc -TERM $XFS
       ;;
    restart)
        $0 stop && $0 start
        ;;
    status|check)
        echo -n "Checking for service $XFS: "
        checkproc $XFS && echo "is running" || echo "is NOT running"
        ;;
    *)
        echo "Usage: $0 {start|stop|status|restart}"
        exit 1
esac

exit 0
```

Also needed are the proper symbolic links in */etc/init.d/rc3.d/* and possibly */etc/init.d/rc5.d*:

```
cd /etc/init.d/rc3.d
ln –s ../xfs S30xfs
ln –s ../xfs K10xfs
cd /etc/init.d/rc5.d
ln -s ../xfs S30xfs
ln -s ../xfs K10xfs
```

This results in the X font server starting every time your Linux system is started.

Adding a server to the font path

Once *xfs* is running on your system, you need to include the font server on your local font path. As before, you can either use *xset* to add the font server for this session, or you can modify *XF86Config* to make the change permanent.

The format of a font-server name is:

```
transport/hostname:port
```

where `transport` refers to the protocol used by the font server. For Linux and Unix environments, the protocol is `tcp` and the default `port` is `7100`. Here is a typical font server name, where the host is a system called *kansas*:

```
tcp/kansas:7100
```

Suppose you're working on the machine *kansas*, and want to access fonts provided by your font server. You just need to add the font server *kansas* to your font path:

```
xset fp+ tcp/kansas:7100
```

Running *xset* with the *q* (query) option shows that Kansas has been added:

```
xset q
```

```
  ...
Font Path:
 /usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/Speedo/,
/usr/X11R6/lib/X11/fonts/75dpi/,tcp/kansas:7100
```

If *xfs* is not running or if you've supplied an incorrect server name, you'll get an error message like the following when you try to add the font server to the path:

```
xset fp+ tcp/kansas:7100
xset:  bad font path element (#38), possible causes are:
    Directory does not exist or has wrong permissions
    Directory missing fonts.dir
    Incorrect font server address or syntax
```

Or, as before, you can make the change permanent by adding the line:

```
FontPath    "tcp/kansas:7100"
```

to your *XF86Config* file to tell the X server to use the X font server when searching for a font.

Once you've added the font server to your font search path, and restarted the X server if necessary, you can specify a font supplied by your X font server on the command line or in a resource file.

Serving TrueType Fonts

One time when you'll definitely want to use a font server is for using TrueType fonts, which are now supported by XFree86 4.0. There are good reasons for wanting to use the TrueType fonts. For one thing, if you are running Linux on a PC or a Macintosh, your system already has TrueType fonts on it, and you might want to use them to extend the range of font options. Also, TrueType fonts were specifically designed for use on computer displays, so they provide a clean, easily readable display. They are scalable, making them easier to see at small sizes.

If you do not have XFree86 4.0 installed, there are still ways you can use TrueType fonts; it's just slightly more complicated. This section describes two options--*xfsft* and *xfstt*--in some detail. It also briefly mentions a third choice, *X-TT*.

xfsft

xfsft is a set of patches to the X font server written by Juliusz Chroboczek to provide TrueType support for X. *xfsft* is based on the FreeType library, which is a free TrueType font rasterizer. (If you want to know more about the FreeType project, see their web page at *http://www.freetype.org*.)

To use *xfsft*, you need to have applied the patches and you need a directory containing your TrueType fonts. After that, you need to create *fonts.scale* and *fonts.dir* files in the TrueType directory, and you need to tell the font server where to find the fonts. The final step is to stop the font server if it is running and restart it with the new TrueType support, or simply to start it if it isn't already running.

If your Linux distribution is Red Hat 6.0 or greater, or a distribution based on Red Hat, you already have the *xfsft* TrueType support on your system. If this was a new installation, you just need to add some TrueType fonts to your system, create the *fonts.scale* and *fonts.dir* files, update the path in */etc/X11/fs/config*, and restart *xfs* to make the TrueType fonts available.

If your installation was an upgrade from an earlier version of Red Hat to Red Hat 6.x, you also need to edit the */etc/X11/XF86Config* file, replacing the existing FontPath entries with the single entry:

```
FontPath "unix/:-1"
```

This differs from the standard *xfs* entry in using the port -1 rather than the default 7100. Red Hat changed the port to -1, which is an invalid port number, for security reasons. This means that your fonts can only be served locally, not across a network.

Red Hat 7.x has gone back to using port 7100, so the FontPath entry in that case is:

```
FontPath "unix/:7100"
```

Red Hat also comes with the *chkfontpath* command. Use the command:

```
chkfontpath −−add directory
```

to add a valid font directory to the *xfs* catalogue entry of font directories.

If the *xfsft* patches aren't available on your system, you can get them from the *xfsft* home page at *http://www.dcs.ed.ac.uk/home/jec/programs/xfsft*, which also has links to sites that make various binary versions available. You'll also want to get the *ttmkfdir* command, created by Jörg Pommnitz and available from his website (*http://www.joerg-pommnitz.de/TrueType/xfsft.html*) . *ttmkfdir* creates the *fonts.scale* file in the TrueType font directory, so you don't have to create it manually.

xfstt

xfstt, or the X11 Font Server for TrueType Fonts, is an alternative to using *xfsft* for serving TrueType fonts. Unlike *xfsft*, *xfstt* doesn't use *xfs*; it is a standalone server for TrueType fonts. You might want to install *xfstt* if you aren't running *xfs* (although you can run both if you want). You also might want to consider *xfstt* if your distribution didn't include the *xfsft* patches, so you can simply install it without having to install patches to *xfs*.

To use *xfstt*, you need to have a directory that contains your TrueType fonts (or symbolic links to the fonts). The default directory is */usr/share/fonts/truetype*. The trick with *xfstt* is to be sure that it is running before you start the X server. To run *xfstt*, do the following:

```
xfstt --sync
xfstt &
```

The first line sets up the *xfstt* database to recognize the TrueType fonts. If your TrueType fonts are somewhere other than the default directory, add the *– –dir* option to specify the directory name:

```
xfstt --dir dir --sync
```

The second line actually starts execution of *xfstt*. Then wait several seconds to be sure it is running, and then you can start X. If you run XDM, you may need to create an *init* file to run *xfstt* before starting X. Or better yet, add these commands to your */sbin/init.d/xfs* file as previously seen in Example 4-2.

You also need to let X know about the font server. You can run *xset fp+* at each login to do that, or you can set it in *XF86Config*. The default *xfstt* port is 7101 (so it doesn't conflict with *xfs*, which uses 7100, in case you want to run both). For example:

```
xset fp+ unix/:7101
```

to add it once you've logged in, or

```
FontPath "unix/:7101"
```

to add it to *XF86Config*.

You can get a usage message with a complete list of options with the *– –help* option:

```
xfstt –help
Xfstt 1.1, X font server for TT fonts
Usage: xfstt [[--gslist]--sync][--port portno][--unstrap][--user username]
             [--dir ttfdir][--encoding list_of_encodings][--daemon][--inetd]
```

xfstt can be downloaded from *ftp://ibiblio.org/pub/Linux/X11/fonts/*.

X-TT

X-TT (the X TrueType Server) is another alternative for using TrueType fonts with X. Like *xfsft*, X-TT uses the Freetype library and is distributed as a set of patches to XFree86; it was designed primarily for use with Asian (CJKV, or Chinese, Japanese, Korean, Vietnamese) fonts, but it supports other character sets as well. X-TT is available at *http://x-tt.dsl.gr.jp/*.

Managing Fonts

The standard Linux system includes many fonts. But it is often the case that "many" is not enough. Whether you want certain fonts for your StarOffice documents or fonts to make the GIMP or Netscape more comfortable to use, you're likely to need to install and manage new fonts. Only fonts that have been correctly installed are used by your X server, which then provides them to the X clients.

Adding and Removing Fonts

As previously explained in the section "Modifying the font search path", you either have to modify *XF86Config* or use the *xset* command to use your new fonts. In addition, before these fonts can be used, they must be put into an X server-recognizable format.

Type 1 fonts

Adobe developed the Type 1 font specification, and produces and distributes many Type 1 typefaces. The Type 1 font specification was originally proprietary, but Adobe released it to third-party font manufacturers with the requirement that all Type 1 fonts adhere to it.

In the section "The fonts.dir File", we mentioned the *fonts.scale* file as an intermediary file for scalable fonts such as Type 1 fonts. If you add Type 1 fonts to your system and they don't come with a *fonts.scale* file, you can use the utility program *type1inst* (available from any of the major Linux websites) to create one. To use *type1inst*, go to the directory containing your fonts and type:

```
type1inst
```

By default, *type1inst* creates a *fonts.scale* file and also a *Fontmap* file used by Ghostscript. It can also be used to generate samples of each of the fonts in the directories, by running it with the *–samples* option. With this option, the command creates a *samples* subdirectory and generates a PostScript file for each Type 1 font. Other options let you run *type1inst* without generating *fonts.scale* (*–nox*), without generating the Ghostscript Fontmap (*–nogs*), or without writing a log file (*–nolog*). Here's an example that only creates the sample files in the *sample* directory:

```
type1inst -nox -nogs -nolog -samples
type1inst Version 0.6.1 (11th February 1998)
Copyright (C) 1996-1998 James Macnicol (james.macnicol@mailexcite.com)

There are a total of 16 PostScript fonts in this directory
[10]
-------------------------------------------------------
16 fonts found
16 were standard PostScript fonts
ls samples
CharterBT-Bold.ps         Courier-Italic.ps             Utopia-Bold.ps
CharterBT-BoldItalic.ps   Courier.ps                    Utopia-BoldItalic.ps
CharterBT-Italic.ps       Courier10PitchBT-Bold.ps      Utopia-Italic.ps
CharterBT-Roman.ps        Courier10PitchBT-BoldItalic.ps Utopia-Regular.ps
Courier-Bold.ps           Courier10PitchBT-Italic.ps    allfont-0.ps
Courier-BoldItalic.ps     Courier10PitchBT-Roman.ps
```

Listing Available Fonts

There are two commands for listing fonts. The first, *.xlsfonts*, lists local fonts, while the command *fslsfonts* lists the fonts supplied by a font server, whether local or remote.

Listing local fonts with xlsfonts

The *xlsfonts* client displays the names of the fonts available locally to your server. When you run *xlsfonts*, it responds with a list that contains the names (including aliases) of the available fonts. By default, *xlsfonts* lists the names of all available fonts. The *–fn* option lets you specify a font name pattern for *xlsfonts* to match; for example, the following finds the URW Courier fonts:

```
xlsfonts -fn '-urw-courier*'
-urw-courier-bold-o-normal--0-0-0-0-p-0-iso8859-1
-urw-courier-bold-r-normal--0-0-0-0-p-0-iso8859-1
-urw-courier-medium-o-normal--0-0-0-0-p-0-iso8859-1
-urw-courier-medium-r-normal--0-0-0-0-p-0-iso8859-1
```

In addition to some options that specify the formatting of the output, there are list options that let you specify various amounts of information for *xlsfonts* to provide. The option *–l* gives a one-line listing for each font that includes some font attributes in addition to the font name. The *–ll* and *–lll* options both provide multiline listings. *–ll* prints font properties in addition to the output provided by *–l*, and *–lll* also lists the character metrics for the font. However, if you are going to use these options, it's wise to narrow down the selection with *–fn* first; using the list options on all the fonts can seriously tie up your system.

Listing fonts on a font server with fslsfonts

If you're running *xfs*, you can use the *fslsfonts* command to list the available fonts. *fslsfonts* lists the names of all fonts available from a specified font server, as in the following example:

```
fslsfonts -server tcp/kansas:7100
```

Since the output is sizeable, it's a good idea to pipe it to a paging program like *more*:

```
fslsfonts -server tcp/kansas:7100 | more
```

You can also limit your search to a particular font (or alias) or use wildcards to search for a group of fonts with the *–fn* option:

```
fslsfonts -server tcp/kansas:7100 -fn '*normal--18*'
-misc-fixed-bold-r-normal--18-120-100-100-c-90-iso10646-1
-misc-fixed-bold-r-normal--18-120-100-100-c-90-iso8859-1
-misc-fixed-medium-r-normal--18-120-100-100-c-90-iso10646-1
-misc-fixed-medium-r-normal--18-120-100-100-c-90-iso8859-1
-misc-fixed-medium-r-normal--18-170-75-75-c-90-iso8859-1
```

The program responds with the names of all font files that matched your specification--in this case, it tells you that the `'*normal--18*'` font is available five times from the font server on *kansas*.

If you get an error such as:

```
fslsfonts: pattern "huh" unmatched
```

it's probably because the font you are looking for isn't available on the server. It's also possible that the *config* file on the server doesn't include all the font directories, so if you're sure the font is there check to make sure you specified it correctly and that the *config* file includes the paths to all font directories.

fslsfonts needs to know what font server to check for fonts. You can either pass it the *–server* option as shown in the example above, or you can set the environment variable FONTSERVER, probably in your X initialization file (e.g.,*.xsession*).

fslsfonts takes other options as well--some let you print various font attributes and properties, and others are options for formatting the output. See the manpage for details of these options.

fsinfo

The *fsinfo* client verifies that a server is running and lists alternative servers (in this case, none):

```
fsinfo -server tcp/kansas:7100
name of server: tcp/kansas:7100
version number: 2
vendor string:  The Open Group
vendor release number:  6300
maximum request size:   16384 longwords (65536 bytes)
number of catalogues:   1
        all
```

```
Number of alternate servers: 0
number of extensions:   0
```

Like *fslsfonts*, *fsinfo* expects the name of a server to be passed as an option, or for the environment variable FONTSERVER to contain the information.

showfont

The *showfont* client provides some rather arcane information about a font served by a font server (much of which is provided for other fonts by *xfd*). *showfont* also provides an ASCII representation of each character in a font. (You can convert the ASCII character into a bitmap image using the *atobm* program if you want to use it as a graphic. See Chapter 3 for instructions.)

To gather data about a font from a server, you can use a command line similar to the following:

```
showfont -server tcp/kansas:7100 -fn fixed > fixed.info
```

In this example, we've redirected the information to a file, *fixed.info*. You might instead pipe it to a paging program, as in the *fslsfonts* example.

Changing Fonts in an xterm Window

xterm includes a `VT Fonts` menu (accessed with the CTRL-button 3 combination) that allows you to change fonts on the fly. Although Chapter 2, *Getting Started Using X*, discussed most of the menu entries, the `Escape Sequence` and `Selection` options required a greater understanding of font naming than we'd covered at that point. Now it's time to describe those options.

Using the escape sequence to pick a font

If you just look at the `VT Fonts` menu in an *xterm* window, you see that the escape sequence entry is grayed out--it's not available. You can make it available by sending an escape sequence plus a new font name to the terminal window. This changes the font and the `Escape Sequence` item on the `VT Fonts` menu becomes available. After that, choosing `Escape Sequence` toggles between the new font and the original *xterm* font.

You send an escape sequence to the *xterm* with the *echo* command. The complete escape sequence to change the *xterm* display font is:

```
Esc]50; fontname CTRL-G
```

In words, the sequence is: the Escape key, the right bracket (`]`), the number 50, a semicolon (`;`), a *`fontname`*, and the CTRL-G key combination. To supply this sequence as an argument to *echo*, enclose it in quotes:

```
echo "Esc]50; fontname CTRL-G"
```

Be aware that when you type the keys as specified, the command line won't look exactly like this on your screen; Escape and CTRL-G will be represented by other symbols on the command line and you'll see something that looks like this:

```
echo "^[]50;fontname^G"
```

You can use a full fontname, an alias, or a wildcarded font specification as the font name. If you use wildcards and the specification matches more than one font, you will get the first font in the search path that matches. The advantage of being able to change the display font with an escape sequence is that it allows you to add another font to the menu choices on the fly for the current *xterm* window. You can change the font associated with the `Escape Sequence` option as often as you want, simply by retyping the escape sequence with a different font name.

The selection menu item

The `Selection` menu item allows you to toggle a font whose name you've previously "selected." For example, you can run *xlsfonts* to list the available fonts, then select the one you want (using the "cut" part of the "cut-and-paste" technique), go to the window whose font you want to change, and select the `Selection` option from the `VT Fonts` menu.

It is more likely, though, that you would use this menu item after selecting a font with *xfontsel*--the `Selection` option was clearly designed with *xfontsel* in mind. If no text is currently selected, the `Selection` menu item is grayed-out, so you can't select it.

The main limitation of `Selection` is that it uses the *last text selected* as the font name, regardless of what that text is. If you select a font name, that name is available through `Selection` only until you use the pointer to select some other text. Since cutting and pasting text is one of the most useful features of *xterm*, you'll probably be making frequent selections. If the most recently selected text is not a valid font name, toggling `Selection` does not change the display font, and X beeps to notify you that the selection failed.

Using International Fonts

We've seen already that X is well-suited for using international fonts--the font-naming conventions were designed to accommodate different character sets; X is distributed with fonts for languages other than English; even the standard English fonts are based on the ISO 8859-1 character set, which supports most of the Western European languages; and many other fonts are available for other character sets.

This section describes the use of international fonts with X, but it also is a convenient place to talk more generally about localizing your X system. First, here are a few useful definitions:

Internationalization
: Internationalization involves writing software that supports multiple languages and nationalities without recompilation. You should be able to add support for a new language to an internationalized system without having to modify the core software. Internationalization is often referred to as the acronym i18n, because there are 18 letters between internationalization's `i` and its `n`.

Localization
: Localization (or L10n because there are ten letters between localization's `l` and its `n`, `L` used to distinguish it from the digit `1`) is the process of setting up an application to be aware of local language and conventions.

Locale

The settings for a particular language and cultural environment, such as the correct currency, the time zone, date and number formatting, and the translation of messages into the local language.

Character Sets

A character set, or more precisely a coded character set, is defined first, by the numerical range of codes; second, by the repertoire of characters; and third, by a mapping between the two sets. As usually the case, a complicated definition means that there many standards that define the many character sets.

7-Bit ASCII

The so-called 7-bit ASCII, or US ASCII, encoding is equivalent to the international standard named ISO-846 established by the International Organization for Standardization (ISO). This encoding actually uses octets of 8 bits per character but leaves the first (the most significant) bit in each octet unused. The remaining 7 bits are capable of encoding the total of 128 ASCII characters. Even though the first 32 codes of ISO 646 are reserved for control characters, which means that they invoke some function or feature in the device that reads and/or displays the text rather than produce a visible shape of a character for human readers, this character set suffices for the English and Latin languages.

ISO 8859

The ISO 8859 standard includes several 8-bit extensions to the ASCII character set (also known as ISO 646-IRV). The full set of ISO 8859 alphabets include:

ISO 8859-1	West European languages (Latin-1)
ISO 8859-2	Central and East European languages (Latin-2)
ISO 8859-3	Southeast European and miscellaneous languages (Latin-3)
ISO 8859-4	Scandinavian/Baltic languages (Latin-4)
ISO 8859-5	Latin/Cyrillic
ISO 8859-6	Latin/Arabic
ISO 8859-7	Latin/Greek
ISO 8859-8	Latin/Hebrew
ISO 8859-9	Latin-1 modification for Turkish (Latin-5)
ISO 8859-10	Lappish/Nordic/Eskimo languages (Latin-6)
ISO 8859-11	Thai
ISO 8859-13	Baltic Rim languages (Latin-7)
ISO 8859-14	Celtic (Latin-8)
ISO 8859-15	West European languages (Latin-9)

ISO 8859-1 and ISO 8859-15 are covered in more detail in the following two sections.

ISO 8859-1

The ISO 8859 standard includes several 8-bit extensions to the ASCII character set. Especially important is ISO 8859-1, the “Latin Alphabet No. 1”, which has become widely implemented and may already be seen as the de-facto standard ASCII replacement.

ISO 8859-1 supports the following languages: Afrikaans, Basque, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Scottish, Spanish, and Swedish.

ISO 8859-15

The ISO 8859-1 character set described in the preceding section lacks the Euro symbol and does not fully cover Finnish and French. ISO 8859-15 is a modification of ISO 8859-1 that covers these needs. Rarely used characters were replaced with these characters.

Unicode

People in different countries use different characters to represent the words of their native languages. Most applications, including email systems and web browsers, are 8-bit clean, i.e., they can operate on and display text correctly provided that it is represented in an 8-bit character set, like ISO-8859-1.

But there are far more than 256 characters in the world--think of Cyrillic, Hebrew, Arabic, Chinese, Japanese, and other Asian languages. The problems that come up for users are:

- It is impossible to store text with characters from different character sets in the same document. For example, you cannot cite Russian papers in a German or French publication using plain text.
- As long as every document has its own character set, and recognition of the character set is not automatic, manual user intervention is needed.
- New symbols such as the Euro currency symbol are constantly being invented. ISO has issued a new standard, ISO-8859-15. If users adopt this standard, they will have documents in different character sets on their disk. But computers should make things simpler, not more complicated.

One solution to this problem is the adoption of a world-wide character set. That character set is Unicode.[*] There are basically four ways to encode Unicode characters into bytes:

UTF-8
: 128 characters are encoded using 1 byte (the ASCII characters). 1920 characters are encoded using 2 bytes (Roman, Greek, Cyrillic, Coptic, Armenian, Hebrew, Arabic characters). 63488 characters are encoded using 3 bytes (Chinese and Japanese among others). The other 2147418112 characters (not assigned yet) can be encoded using 4, 5, or 6 characters.

[*] For more information about Unicode, see the Unicode HOWTO, *http://www.unicode.org*, or issue the command `man -7 unicode`.

UCS-2

Every character is represented as two bytes. This encoding can only represent the first 65536 Unicode characters.

UTF-16

This is an extension of UCS-2 that can represent 1114112 Unicode characters. The first 65536 Unicode characters are represented as two bytes, the rest as four bytes.

UCS-4

Every character is represented as four bytes.

The space requirements for encoding a text compared to encodings currently in use (8 bits per character for European languages, more for other languages) is as follows. This has an influence on disk storage space and network download speed.

UTF-8

Uses no additional space for US ASCII, only slightly more for ISO-8859-1, 50% more for Chinese/Japanese/Korean, and 100% more for Greek and Cyrillic.

UTS-2 and UTF-16

Uses no additional space for Chinese/Japanese/Korean, 100% more for US ASCII and ISO-8859-1, Greek and Cyrillic.

UCS-4

Uses 100% more space for Chinese/Japanese/Korean, 300% more for US ASCII and ISO-8859-1, Greek and Cyrillic.

Given the penalty for US and European documents caused by UCS-2, UTF-16, and UCS-4, it seems unlikely that these encodings have a potential for wide-scale use. UTF-8 on the other hand has the potential for wide-scale use since it does not penalize US and European users.

Keyboard

When you set up your X system, you were asked to specify your keyboard's model and language. Your responses are reflected in the XFree86 configuration file, *XF86Config*, as in the following example:

```
Section "Keyboard"
  Protocol "Standard"
  XkbRules "xfree86"
  XkbModel "pc102"
  XkbLayout       "de_CH"
  XkbVariant      "nodeadkeys"
  XkbOptions      "ctrl:ctrl_aa"
EndSection
```

It's important to note the `XkbLayout` entry in the `Keyboard` section. You can find the valid values for `XkbLayout` in the directory */usr/X11R6/lib/X11/xkb/symbols/*. XFree86 gets its information for mapping the keycode to the correct character from these keyboard description files. In the example, the `de_CH` entry tells XFree86 that we are using a Swiss-German keyboard. This entry can be manually changed.

Table 13-4 shows the supported keyboards and the code to use for `XkbLayout`.

Table 13-4: Supported keyboards

Code	**Description**	**Code**	**Description**
be	Belgian	gb	Great Britain
bg	Bulgarian	hu	Hungarian
br	Brazilian	is	Icelandic
ca	Canadian	it	Italian
cs	Czech	no	Norwegian
de	German	pl	Polish
de_CH	Swiss German	pt	Portuguese
dk	Danish	ru	Russian
dvorak	Dvorak	se	Swedish
ee	Estonian	si	Slovene
es	Spanish	th	Thai
fi	Finnish	ua	Ukrainian
fr	French	us	US/ASCII
fr_CH	Swiss French	us_intl	US/ASCII

xmodmap

xmodmap is a utility for modifying key mappings. It has many uses; we saw one example in Chapter 12, "Setting Resources". In this section, we'll look at a more complex example.

A peculiarity of Switzerland is that its French speaking people don't use a French keyboard, nor do the German speaking people use a German keyboard, but the Italian part of Switzerland does use an Italian keyboard! The Swiss French and Swiss German keyboards are so similar that the keyboard manufacturers often only make one Swiss keyboard. However, the differences are clearly marked on the keyboard. Figure 13-10 shows the key that is two keys to the right of the L Key.

[Illustration department, please draw me a key with the following keycap:

```
à ä
ä à {
```

Thanks!]

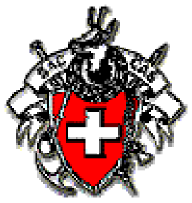


Figure 13-10: A Swiss key

A Swiss German often uses the German letter ä, but rarely the French letter à. Probably the { is used more often than the à. But a Swiss French often uses the à, much more often than the letter ä. So what's the solution if a Swiss German shares a system with a Swiss

French? Rather than constantly reconfiguring the X server, there's a solution provided by *xmodmap*. Its *–pke* parameter prints the currently used keymap table to standard output. If such a valid keymap is used as *xmodmap*'s input, then the X server immediately begins using the new keyboard mapping.

Let's say you are a Swiss German, and you want to set your system up so your Swiss French friend can also use it conveniently. You can perform the following 6 steps once, and after that your friend will only have to follow the simple procedure described in the paragraph that follows the list of 6 steps.

1. Start your X Window system and open a valid terminal window (e.g., *xterm*).
2. Save the current keyboard configuration (in this case Swiss German) in your home directory with the following command:

```
xmodmap –pke > ~/Xmodmap.de_CH
```

3. Reconfigure the X Window system by editing your *XF86Config* file for the second keyboard you'd like to use (in this example Swiss French). This is now the new default keymap.
4. Restart the X Window system and again open a terminal window.
5. Save the active (new) keymap:

```
xmodmap –pke > ~/Xmodmap.fr_CH
```

6. Reconfigure and restart your X Window system once more for your default Swiss German keymap.

From now on, to change the keymap without reconfiguring and restarting the X server, simply call *xmodmap* in a terminal window with the desired keymap as its only parameter:

```
xmodmap ~/Xmodmap.fr_CH
```

In this case we've changed the keyboard mapping to the Swiss French keyboard mapping that we saved earlier. If you're Swiss French, and you are constantly working on your Swiss German friend's system, you can place this command in your startup file rather than running it manually every time. Note that etiquette requires you to change the keyboard back to Swiss German by running the following command before you log out:

```
xmodmap ~/Xmodmap.de_CH
```

Fonts

Most fonts in the standard X distribution use the ISO 8859-1 character set. This is reflected in the fontname, which has *iso8859* in the registry portion of the name and *1* in the decoding portion. This ISO Latin-1 character set is a superset of the standard ASCII character set.

Use *xlsfonts* or *fslsfonts* as described in the "Listing available fonts" section of this chapter to see whether the font you're presently using supports your preferred character set. For example, you can see if it supports the Euro symbol by searching *xlsfonts*'s output for `iso8859-15`:

```
xlsfonts | uniq | grep –i iso8859-15
-b&h-lucidux mono-medium-o-normal--0-0-0-0-m-0-iso8859-15
-b&h-lucidux mono-medium-r-normal--0-0-0-0-m-0-iso8859-15
-b&h-lucidux sans-medium-o-normal--0-0-0-0-p-0-iso8859-15
```

```
-b&h-lucidux sans-medium-r-normal--0-0-0-0-p-0-iso8859-15
-b&h-lucidux serif-medium-o-normal--0-0-0-0-p-0-iso8859-15
-b&h-lucidux serif-medium-r-normal--0-0-0-0-p-0-iso8859-15
-misc-fixed-bold-r-normal--0-0-75-75-c-0-iso8859-15
-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-15
-misc-fixed-medium-r-normal--0-0-75-75-c-0-iso8859-15
-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-15
9x15
```

If the characters you need aren't found in the fonts available to you, you may need to download and install fonts as described in the many HOWTOs mentioned in the next section.

Language HOWTOs

There are HOWTOs for a number of languages, some written in English, some in the native language. The following table lists the language HOWTOs and indicates if they are in the native language:

HOWTO	**Native?**	**HOWTO**	**Native?**	**HOWTO**	**Native?**
Belgian		French	X	Portuguese	X
Chinese		German	X	Serbian	X
Cyrillic		Hebrew		Slovenian	X
Danish		Hellenic	X	Spanish	X
Esperanto	X	Italian	X	Thai	
Finnish	X	Polish	X	Turkish	X

If you don't find a HOWTO for the language you want in your distribution (check */usr/doc/HOWTO*), you can check the Linux Documentation Project website at *http://www.linuxdoc.org*. You also might want to check some of the other language HOWTOs, since many of the issues are the same for all languages--you'll probably get enough information to be able to extrapolate to your language. (Once you have it figured out and if you're feeling generous, you might want to write the HOWTO yourself and contribute it!)